

GPU accelerated time domain discrete geometric approach method for Maxwell's equations on tetrahedral grids

Matteo Cicuttin^{1,2}, Lorenzo Codecasa³, Bernard Kapidani⁴, Ruben Specogna⁴, Francesco Trevisan⁴

¹Université Paris-Est, CERMICS (ENPC), F-77455 Marne-la-Vallée, France

²INRIA Paris, 2 rue Simone Iff, F-75589 Paris, France

³Politecnico di Milano, Dip. di Elettronica, Informazione e Bioingegneria, I-20133, Milano, Italy

⁴Polytechnic Department of Engineering and Architecture (DPIA), Università di Udine, I-33100 Udine, Italy

A recently introduced time domain method for the numerical solution of Maxwell's equations on unstructured grids is reformulated in a novel way, with the aim of implementation on Graphical Processing Units (GPUs). Numerical tests show that the GPU implementation of the resulting scheme yields correct results, while also offering an order of magnitude in speedup and still preserving all the main properties of the original Finite Difference Time Domain (FDTD) algorithm.

Index Terms—Discrete Geometric Approach, Wave Propagation, Parallel computing, FDTD

I. INTRODUCTION

THE extension of Yee's original FDTD algorithm [1] to unstructured grids is a topic of intense research in computational electromagnetics. Unstructured grids, differently from the cartesian grids used in the original FDTD, allow to easily handle the complex geometries arising in applications from both experimental science and industry. However, to be successful, FDTD-like algorithms on unstructured grids must retain the properties that make the original algorithm so ubiquitous. One of these is its uniform and small stencil, which makes the idea of a numerical time domain solution of huge wave propagation problems on machines with parallel computing capabilities very appealing. In practice, the algorithm has thus seen its performance bottleneck getting more and more linked to the memory bandwidth of the underlying hardware architecture. For this reason the focus of FDTD code developers has recently shifted from multicore CPUs to Graphical Processing Units (GPUs) [3], [4], which provide typically an order of magnitude more bandwidth than high performance CPUs for the price of a medium budget laptop.

On unstructured grids, a parallel implementation using a cluster of GPUs has been presented in [5] based on the time domain application of the Discontinuous Galerkin (DG) Finite Element Method. However, DG formulations exhibit spurious mode solutions due to violation of charge (and possibly energy) conservation [6].

In the present paper we will use the approach of [7], based on the Discrete Geometric Approach (DGA), which yields a conditionally stable, explicit and consistent scheme on tetrahedral grids. Furthermore the algorithm, to which we will refer as the DGATD in the following, is stencil-based as the original FDTD. The accuracy and performance of this approach on single-core CPU architectures have already been

studied in [8]. The aim of this paper is instead to present a reformulation of this approach which also exhibits the same promise for exploiting hardware parallelization as the original FDTD method.

The paper is organized as follows: in Section II the space and time discretization approach is outlined in detail, in Section III a brief stability analysis for the resulting numerical scheme is given, in Section IV the correctness of the implementation is assessed on a test case with known analytical solution, in Section V implementation and performance related details within the NVIDIA Compute Unified Device Architecture (CUDA) programming model are given. Finally, in Section VI, some conclusions are drawn.

II. FRACTIONED GRID FORMULATION

We wish to solve the initial value problem for Maxwell's equations

$$\mu \frac{\partial \mathbf{h}}{\partial t} = -\nabla \times \mathbf{e}, \quad (1)$$

$$\varepsilon \frac{\partial \mathbf{e}}{\partial t} = \nabla \times \mathbf{h}, \quad (2)$$

in a bounded region $\Omega \subset \mathbb{R}^3$, in which we assume no conductive losses. In the framework of the Discrete Geometric Approach (DGA), electromagnetic quantities are discretized into degrees of freedom (DoFs) by their integrals over geometric elements of two dual grids: a primal tetrahedral mesh \mathcal{G} and a polyhedral dual mesh (dual complex) $\tilde{\mathcal{G}}$ obtained by barycentric subdivision of \mathcal{G} . In the DGA, discrete differential operators are encoded by the incidence matrices between geometric elements in each mesh: in (1) the flux of the curl of \mathbf{e} through any triangle of \mathcal{G} equals the (oriented) sum of the circulations of \mathbf{e} over the edges in its boundary; similarly in (2) the flux of the curl of \mathbf{h} through any face of $\tilde{\mathcal{G}}$ equals the (oriented) sum of the circulations of \mathbf{h} over the dual edges in its boundary. A remarkable consequence is that for Maxwell's equations written in integral form the curl operator discretization is exact.

Manuscript received June ??, 2017; accepted ??, 2017. Date of current version June 27, 2017. Corresponding author: B. Kapidani (e-mail: kapidani.bernard@spes.uniud.it).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 00.0000/TMAG.2017.0000000

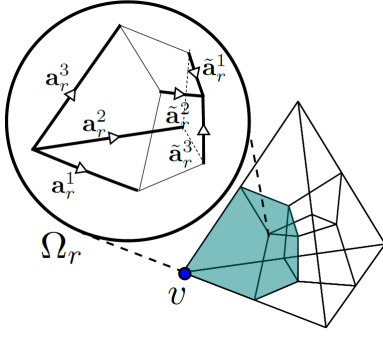


Fig. 1. Hexahedron Ω_r is the element on which the basis functions are defined.

Let us take any disjoint intersection Ω_r between a tetrahedron τ_n of \mathcal{G} and a dual volume $\tilde{\tau}_v$ of \mathcal{G} centered in any of the four vertices v of τ_n (see Fig. 1). By construction, Ω_r is always a hexahedron. Furthermore, τ_n is the union of four disjoint hexahedra, while $\tilde{\tau}_v$ is the union of a variable number of disjoint hexahedra (equal to the number of tetrahedra in which v is a vertex). We will define fractioned DoFs in each Ω_r as

$$\hat{\mathbf{f}}_r = \begin{pmatrix} \mathbf{h} \cdot \tilde{\mathbf{a}}_r^1 \\ \mathbf{h} \cdot \tilde{\mathbf{a}}_r^2 \\ \mathbf{h} \cdot \tilde{\mathbf{a}}_r^3 \end{pmatrix}, \quad \hat{\mathbf{u}}_r = \begin{pmatrix} \mathbf{e} \cdot \mathbf{a}_r^1 \\ \mathbf{e} \cdot \mathbf{a}_r^2 \\ \mathbf{e} \cdot \mathbf{a}_r^3 \end{pmatrix},$$

where $\tilde{\mathbf{a}}_r^1, \tilde{\mathbf{a}}_r^2, \tilde{\mathbf{a}}_r^3$ are the tangent vectors of the portions of the three edges of $\tilde{\tau}_v$ in the boundary of Ω_r (as oriented in Fig. 1) and $\mathbf{a}_r^1, \mathbf{a}_r^2, \mathbf{a}_r^3$ are the tangent vectors of the portions of the three edges of τ_n in the boundary of Ω_r (as oriented in Fig. 1). These DoFs are the main difference with respect to [7] and will lead to a different, although equivalent, structure in the resulting algorithm. Furthermore we define a local face-edge incidence matrix in Ω_r

$$\mathbf{C}_r = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix},$$

and two distinct sets of uniform valued basis functions $\tilde{\mathbf{w}}_r^i(\mathbf{r})$ and $\mathbf{w}_r^i(\mathbf{r})$, with $i = 1, 2, 3$, which have compact support and can be written as

$$\tilde{\mathbf{w}}_r^i(\mathbf{r}) = \frac{\tilde{\mathbf{a}}_r^j \times \tilde{\mathbf{a}}_r^k}{\tilde{\mathbf{a}}_r^i \times \tilde{\mathbf{a}}_r^j \cdot \tilde{\mathbf{a}}_r^k}, \quad (3)$$

$$\mathbf{w}_r^i(\mathbf{r}) = \frac{\mathbf{a}_r^j \times \mathbf{a}_r^k}{\mathbf{a}_r^i \times \mathbf{a}_r^j \cdot \mathbf{a}_r^k}, \quad (4)$$

in which i, j, k is any permutation of 1, 2, 3. Finally we define two local mass matrices, whose element at the i -th row and j -th column is

$$\mathbf{M}_r^\mu(i, j) = \int_{\Omega_r} \tilde{\mathbf{w}}_r^i(\mathbf{r}) \cdot \mu(\mathbf{r}) \tilde{\mathbf{w}}_r^j(\mathbf{r}) d\mathbf{r},$$

$$\mathbf{M}_r^\varepsilon(i, j) = \int_{\Omega_r} \mathbf{w}_r^i(\mathbf{r}) \cdot \varepsilon(\mathbf{r}) \mathbf{w}_r^j(\mathbf{r}) d\mathbf{r}.$$

Since \mathbf{M}_r^μ and \mathbf{M}_r^ε are 3×3 symmetric, positive-definite matrices we are easily tempted to discretize Eqs. (1)–(2) in each Ω_r . To impose continuity of tangential field components

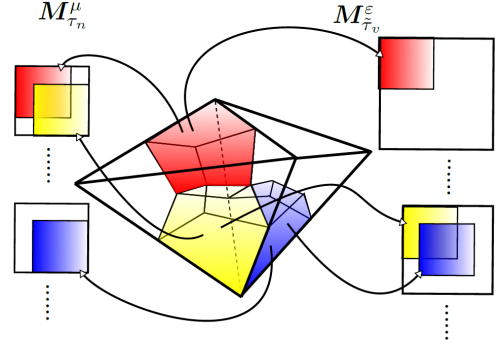


Fig. 2. Procedure of assembling local mass matrices: detail of two adjacent tetrahedra.

as in the original FDTD, we proceed thinking in global variables terms: the set of unknowns of the discrete problem comprises the set $\hat{\mathbf{u}}$, i.e. the circulations of \mathbf{e} over *each half of each edge* of \mathcal{G} without repetitions, and the set $\hat{\mathbf{f}}$, i.e. the circulations of \mathbf{h} over *each half of each edge* of $\tilde{\mathcal{G}}$, again without repetitions (we split a full dual edge at the barycenter of the triangle it crosses). To get a consistent discretization of (1), we have to glue together the contributions of the four Ω_r -like hexahedra in τ_n . Take for example the simple case of Fig. 2 in which \mathcal{G} is just two tetrahedra: the yellow and the red hexahedra are in the same tetrahedron, so their local $\mathbf{M}_{\tau_n}^\mu$ can be assembled to contribute to the local mass matrix of a full tetrahedron. If we do likewise for the other two Ω_r -like hexahedra in the same tetrahedron we get

$$\mathbf{M}_{\tau_n}^\mu \frac{\hat{\mathbf{f}}_{\tau_n}^{n+\frac{1}{2}} - \hat{\mathbf{f}}_{\tau_n}^{n-\frac{1}{2}}}{\Delta t} = -\mathbf{C}_{\tau_n} \hat{\mathbf{u}}^n,$$

where $\mathbf{M}_{\tau_n}^\mu$ is a 4×4 matrix and $\hat{\mathbf{f}}_{\tau_n}$ is a column vector of size 4. Matrix \mathbf{C}_{τ_n} is a (very sparse) superposition of the appropriate \mathbf{C}_r matrices, and again can be constructed efficiently.

Referring again to Fig. 2, the yellow and the blue hexahedra are instead in two different tetrahedra, therefore they contribute to separate $\mathbf{M}_{\tau_n}^\mu$. Yet these two hexahedra share a vertex v of \mathcal{G} , so they will contribute to a local mass matrix $\mathbf{M}_{\tilde{\tau}_v}^\varepsilon$, associated to the volume of $\tilde{\mathcal{G}}$ centered in v . Then, to get a consistent discretization of (2), we glue together the contributions of all the Ω_r -like hexahedra in a dual $\tilde{\tau}_v$

$$\mathbf{M}_{\tilde{\tau}_v}^\varepsilon \frac{\hat{\mathbf{u}}_{\tilde{\tau}_v}^{n+1} - \hat{\mathbf{u}}_{\tilde{\tau}_v}^n}{\Delta t} = \tilde{\mathbf{C}}_{\tilde{\tau}_v} \hat{\mathbf{f}}^{n+\frac{1}{2}},$$

where $\mathbf{M}_{\tilde{\tau}_v}^\varepsilon$ is a $\tilde{d} \times \tilde{d}$ matrix, $\hat{\mathbf{u}}_{\tilde{\tau}_v}$ is a column vector of size \tilde{d} , where \tilde{d} is the number of edges of \mathcal{G} which contain v and $|v|$ is the number of tetrahedra of \mathcal{G} which contain v . Matrix $\tilde{\mathbf{C}}_{\tilde{\tau}_v}$ is a (very sparse) superposition of the appropriate set of \mathbf{C}_r^T matrices and can be constructed efficiently. We note the remarkable fact that, by requiring tangential continuity of \mathbf{e} and \mathbf{h} and exploiting the properties of (3)–(4), matrices $\mathbf{M}_{\tau_n}^\mu$ and $\mathbf{M}_{\tilde{\tau}_v}^\varepsilon$ are locally and globally exactly divergence preserving matrices for piecewise-constant fields $\mathbf{e} = \mathbf{e}_{\tilde{\tau}_v}$ and $\mathbf{h} = \mathbf{h}_{\tau_n}$ (see [7] for the proof). Furthermore they are symmetric positive-definite, small, sparse and can be inverted

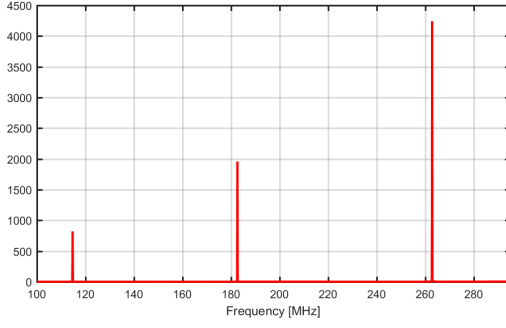


Fig. 3. Result of the FFT on the computed electric field in the cylindrical resonator. The resonances match the theoretically predicted frequency values.

locally, yielding

$$\hat{\mathbf{f}}_{\tau_n}^{n+\frac{1}{2}} = \hat{\mathbf{f}}_{\tau_n}^{n-\frac{1}{2}} - \Delta t (\mathbf{M}_{\tau_n}^{\mu})^{-1} \mathbf{C}_{\tau_n} \hat{\mathbf{u}}^n \quad \forall \tau_n \in \mathcal{G}, \quad (5)$$

$$\hat{\mathbf{u}}_{\tilde{\tau}_v}^{n+1} = \hat{\mathbf{u}}_{\tilde{\tau}_v}^n + \Delta t (\mathbf{M}_{\tilde{\tau}_v}^{\varepsilon})^{-1} \tilde{\mathbf{C}}_{\tilde{\tau}_v} \hat{\mathbf{f}}^{n+\frac{1}{2}} \quad \forall \tilde{\tau}_v \in \tilde{\mathcal{G}}, \quad (6)$$

which is a valid, explicit, time marching scheme. We remark that the local curl matrices \mathbf{C}_{τ_n} , $\tilde{\mathbf{C}}_{\tilde{\tau}_v}$ are very sparse and their product with global field vectors can be performed in parallel. We will refer to (5)–(6) as the fractioned Discrete Geometric Approach Time Domain (fractioned DGATD) method in the rest of the paper. We remark that (5)–(6) is a reformulation of the scheme of [7], and that the two yield exactly the same solution up to machine precision.

III. STABILITY ANALYSIS

The fractioned DGATD is conditionally stable due to the properties of its mass matrices [7]. Simulations with more than 300 000 time steps do not show any late time instabilities, as also the test case of section IV will show. If we pack the system of equations in (5)–(6) by appending all the local mass matrices into global block-diagonal, matrices $\mathbf{M}_{\tau}^{\mu-1}$, $\mathbf{M}_{\tilde{\tau}}^{\varepsilon-1}$, by appending all the local curl matrices, and by appending all the field unknowns in column vector format, the scheme reads

$$\hat{\mathbf{f}}^{n+\frac{1}{2}} = \hat{\mathbf{f}}^{n-\frac{1}{2}} - \Delta t \mathbf{M}_{\tau}^{\mu-1} \mathbf{C}_{\tau} \hat{\mathbf{u}}^n, \quad (7)$$

$$\hat{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^n + \Delta t \mathbf{M}_{\tilde{\tau}}^{\varepsilon-1} \tilde{\mathbf{C}}_{\tilde{\tau}} \hat{\mathbf{f}}^{n+\frac{1}{2}}, \quad (8)$$

where we have dropped the subscripts on τ and $\tilde{\tau}$ to stress that the constitutive matrices are not restricted any more to a particular primal or dual volume, respectively. Likewise \mathbf{C}_{τ} and $\tilde{\mathbf{C}}_{\tilde{\tau}}$ are obtained by appending all the \mathbf{C}_{τ_n} and $\mathbf{C}_{\tilde{\tau}_v}$, respectively. Eq. (8) can be rewritten as

$$\hat{\mathbf{u}}^{n+1} = 2\hat{\mathbf{u}}^n - \hat{\mathbf{u}}^{n-1} + \Delta t \mathbf{M}_{\tilde{\tau}}^{\varepsilon-1} \tilde{\mathbf{C}}_{\tilde{\tau}} \left(\hat{\mathbf{f}}^{n+\frac{1}{2}} - \hat{\mathbf{f}}^{n-\frac{1}{2}} \right),$$

from which, using (7), it ensues

$$\hat{\mathbf{u}}^{n+1} = \left(2\mathbf{I} - \Delta t^2 \mathbf{M}_{\tilde{\tau}}^{\varepsilon-1} \tilde{\mathbf{C}}_{\tilde{\tau}} \mathbf{M}_{\tau}^{\mu-1} \mathbf{C}_{\tau} \right) \hat{\mathbf{u}}^n - \hat{\mathbf{u}}^{n-1}.$$

The timestepping structure is thus shown to be equivalent to the one proposed on standard Finite Elements in [9]. For a given grid, the fractioned DGATD scheme is stable provided that $\Delta t < 2/\sqrt{\rho_K}$, where ρ_K is the spectral radius of the matrix $\mathbf{K} = \mathbf{M}_{\tilde{\tau}}^{\varepsilon-1} \tilde{\mathbf{C}}_{\tilde{\tau}} \mathbf{M}_{\tau}^{\mu-1} \mathbf{C}_{\tau}$. This means that the maximum

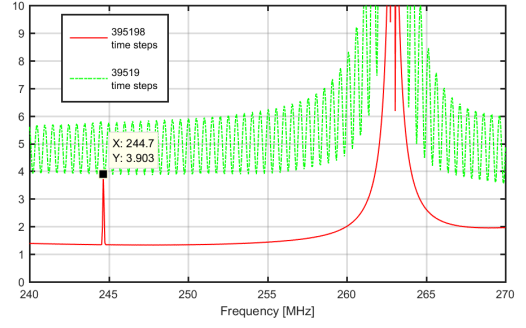


Fig. 4. Zoom in frequency. The theoretically predicted peak at 245.06 MHz is below the noise floor if the simulation is too short.

allowed Δt can be efficiently estimated using algorithms based on the power iteration method.

IV. CORRECTNESS ANALYSIS

The proposed formulation was tested on a problem with known analytical solution inspired from [10]. We tested the algorithm on a cylindrical resonator (height $h = 0.5\text{m}$ and radius $r = 1\text{m}$) filled with air. The resonator is excited via external coupling with a broad Gaussian pulse. Thus we wish to solve an eigenvalue problem in the time domain, since it is known that the resonant frequencies are given by

$$f_{mnp} = \frac{c}{2\pi\sqrt{\mu_r\varepsilon_r}} \sqrt{\left(\frac{\chi_{mn}}{r}\right)^2 + \left(\frac{p\pi}{h}\right)^2},$$

where χ_{mn} is the m -th zero of the n -th cylindrical Bessel function, p is a non-negative integer and c is the speed of light in vacuum. We can retrieve spectral behaviour (Figs. 3,4) for an arbitrarily large frequency interval after a single time domain simulation by computing the FFT on the interpolated e in one element of \mathcal{G} . We know from the analysis that we should see spectral peaks in Fig. 3 at frequencies $f_1 = 114.75\text{ MHz}$, $f_2 = 182.84\text{ MHz}$, $f_3 = 245.06\text{ MHz}$, $f_4 = 263.38\text{ MHz}$. The peaks at f_1 , f_2 , and f_4 are visible in Fig. 3. The peak at f_3 seems to be missing, but is actually retrieved by zooming in the range between 240 and 270 MHz (see Fig. 4). We remark that, to resolve the peak at f_3 from the nearest peak, a rather high sampling frequency is needed, even after applying Hamming windowing on the input waveform. This constraint translates into a longer simulation (a further motivation for exploiting parallelism).

V. PERFORMANCE ANALYSIS

The method was implemented in C++. For the single-core CPU version we used the original DGATD of [7], running on a Xeon E5-2687Wv4 processor with Eigen 3.3.1 for linear algebra operations. The CUDA C++ code was tested on a TESLA C2075 accelerator. Two reasonable approaches are possible in implementing the algorithm on the GPU. The first, more abstract approach, is to still use the original formulation of [7], which features huge sparse matrix vector multiplications as its only conspicuous floating point operation, and adapt it to use the cuSparse library provided by NVIDIA, which provides a sparse matrix vector multiplication (SpMV) function. On the

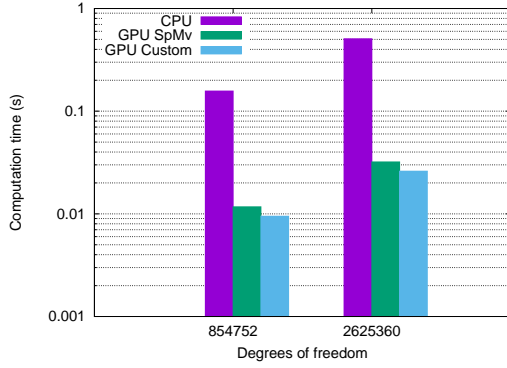


Fig. 5. Average computational cost of a single time step vs. number of DoFs of the problem. x axis in linear scale, y axis in logarithmic scale.

test case described in Section IV, somewhat surprisingly, this rather naive GPU implementation already provides an order of magnitude of speedup with respect to CPU based one, as shown in of Table I. However, SpMV produces memory access patterns that are not optimal on GPUs [11]. This detail was actually the motivation for the reformulation of the algorithm presented in Section II, in which we try to exploit the block-diagonal structure of the mass matrices to improve performance. The key insight is the following: high latency access to the global memory of the GPU is optimized if consecutive threads access consecutive locations of global memory. By looking at the right-hand side of Eq. (5) we are presented with a set of local 4×6 matrices which can be stored using column-major ordering. This allows adjacent matrix values to be loaded in shared memory by adjacent threads, obtaining full memory coalescing. Local matrix vector products are then computed using one thread per row.

TABLE I
SPEEDUPS

DoFs	CPU	SpMV	Cust
854752	1	13.42	16.58
2625360	1	15.84	19.50

Experiments on our accelerator, which provides a memory bandwidth of roughly 102 GB/s, show that a matrix multiplication kernel based on this insight reaches a sustained rate of 98 GB/s. The global speedup achieved with this implementation is shown again in Table I. Unfortunately the same approach does not perform equally well on Eq. (6), as the variable size of the local mass matrix makes the logic to be added trickier and heavier. For the aims of this paper, we decided to settle for an implementation exploiting a specific kernel only for Eq. (5), relying on cuSparse for the remaining computations. In Fig. 5 we show the average single time step computational time versus the number of unknowns of the problem for two different meshes. In Fig. 6 we show the speedup achieved by using the fractioned grid formulation in Eq. (5), with respect to the original formulation of [7]. We remark that the fact that there is a speedup is non-trivial, since by using the fractioned DGATD formulation we are somewhat increasing the number of DoFs in the problem.

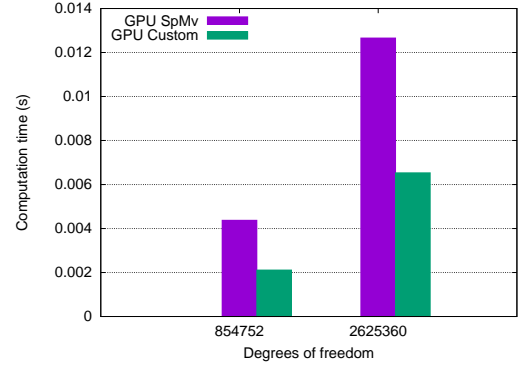


Fig. 6. Average computational cost of a single execution of the kernel for Faraday's law vs. number of DoFs of the problem. Both axes in linear scale.

VI. CONCLUSIONS

We presented a reformulation of the DGATD algorithm on tetrahedral grids. The formulation is amenable to parallelization on a GPU, achieving an order of magnitude of speedup. We are currently investigating improved data layouts to further increase performance; those improvements will be the subject of a subsequent work.

ACKNOWLEDGEMENT

This work was partially supported by the Strategic Plan of the Polytechnic Department of Engineering and Architecture (DPIA) of University of Udine.

REFERENCES

- [1] K.S. Yee, Numerical solution of initial boundary value problems involving Maxwells equations in isotropic media, *IEEE Transactions on Antennas and Propagation*, vol. AP-14, no. 3, pp. 302307, May 1966.
- [2] A. Taflove and S. Hagness, *Computational Electromagnetics, The Finite Difference Time Domain Method*, Second Edition. Boston, MA: Artech House, 2000.
- [3] P. Sypek, A. Dziekonski, and M. Mrozowski, How to Render FDTD Computations More Effective Using a Graphics Accelerator, *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1324-1327, 2009.
- [4] D. De Donno, A. Esposito, L. Tarricone, L. Catarinucci, Introduction to GPU Computing and CUDA Programming: A Case Study on FDTD, *IEEE Antennas and Propagation Magazine*, Vol. 52, No. 3, pp. 116-122, 2010.
- [5] N. Goedel, N. Gunn, T. Warburton, M. Clemens, Scalability of Higher-Order Discontinuous Galerkin FEM Computations for Solving Electromagnetic Wave Propagation Problems on GPU Clusters, *IEEE Transactions on Magnetics*, Vol. 46, No. 8, pp. 3469-3472, 2010.
- [6] E. Gjonai, T. Lau, T. Weiland, Conservation Properties of the Discontinuous Galerkin Method for Maxwell Equations, *International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Torino, Italy, 17-21 September 2007.
- [7] L. Codecasa, M. Politi, Explicit, Consistent and Conditionally Stable Extension of FD-TD by FIT, *IEEE Transactions on Magnetics*, Vol. 44, pp. 1258-1261, 2008.
- [8] B. Kapidani, L. Codecasa, M. Cicuttin, R. Specogna, F. Trevisan, A comparative performance analysis of time-domain formulations for wave propagation problems, *2016 IEEE Conference on Electromagnetic Field Computation (CEFC)*, Miami, FL, 13-16 November 2016.
- [9] J.M. Jin, *The Finite Element Method in Electromagnetics*, Second Edition, Wiley, New York, 2002.
- [10] M. Cinalli, A. Schiavoni, A Stable and Consistent Generalization of the FDTD Technique to Nonorthogonal Unstructured Grids, *IEEE Transactions on Antennas and Propagation*, Vol. 54, No. 5, pp. 1503-1512, 2006.
- [11] N. Bell, M. Garland, Efficient Sparse Matrix-Vector Multiplication on CUDA, *NVIDIA Technical Report NVR-2008-004*, Dec. 2008.